Evolving A Legacy System: Restructuring The Mendelian Inheritance in Man Database

Peter Li, Ph.D., Laurie Kramer, Stuart Pineo, and David Kulp
Genome Database, Johns Hopkins University, Baltimore, MD

Mendelian Inheritance in Man (MIM) is an encyclopedia of medical genetics that has been in electronic form for over 30 years. In its lifetime, MIM has undergone many organizational and software changes. In 1994, a major transition was made based on three basic principles: industry standards, open systems architecture, and extensibility. The resulting MIM database allows users to navigate to other genomic databases, permits the delivery of multimedia information, and improves the quality of data. The new MIM database also improved its administration because of 1) an internal format that enforces consistency; 2) a lower maintenance cost of software; and 3) a better ability to migrate MIM content. In addition, the new architecture will allow MIM easily to adopt emergent technologies as they mature.

INTRODUCTION

MIM is a comprehensive collection of literature reviews for the field of medical genetics [1]. It is authored by Dr. Victor A. McKusick at the Johns Hopkins University, School of Medicine and has been maintained in electronic format since 1963. Initially, entries recorded in the MIM database were mendelizing traits (phenotypes) reported in the literature. However, as the science of genetics evolved, entries about genes were also added to the database. Consequently, MIM began to evolve as a "gene catalog." Currently, there are 6,600 entries. Each entry is assigned a 6-digit number for identification. The total text size is 27 megabytes, 40 percent of which are 52,000 references.

The MIM editorial system, i.e., the repository that supports the editing and distribution of MIM entries, had not been updated for the last 10 years. It only supported three specific UNIX accounts for editing and the data were made available to users through only two media. The first medium is the printed book form, published once every 2 years since 1963. The second medium is the terminal-based online version of MIM (OMIM), using the "Information Retrieval eXperimental Workbench" (IRX) programs developed in collaboration with the National Library of Medicine in 1985 [2].

When compared to other genomic databases [3], many of which are less than 10 years old, the MIM repository was based on outdated principles and technologies. It was a classic example of a legacy system developed without applicable standards, constrained to a centralized machine, and based on proprietary code that could not be migrated easily [4]. Consequently, it was increasingly difficult to connect MIM information with other genomic databases and to provide MIM information through user-friendly means.

In order to maintain MIM's usefulness, it had to evolve to new standards and to meet new expectations. For example, the technology for accessing information through CDROM and Internet have matured and users want to be able to navigate to and from other genomic databases. However, the characteristics of the legacy system makes this evolution very difficult. Different strategies were considered, ranging from encapsulating the MIM system with a shell to migrating the data and software onto new architectures. We chose the latter approach because the encapsulation approach would not sufficiently resolve the limitations of the system.

MIM required a different approach for migration than the traditional databases of the business world because it is a text-oriented database. Text databases are under intense research from the information retrieval community [5][6], in part because of the difficulty of prescribing a single structure for the many unrelated flat files, e.g., as found in news and print archives. However, a subset of these wide-spectrum text databases is characterized by files that come from a single source or relate to a single topic, e.g., bibliographic databases. For these databases, it is easier to determine a single structure (schema) to which all contents can be converted. Once in this structured format, one can maintain consistency and integrity of the entries through database techniques.

MIM is an ideal candidate to evolve to a structured form because it is a collection of text files that comes from a single source and relates to a single topic. There are many approaches to defining structure for documents. One approach is based on the presentation format, such as titles, headers, and paragraphs. Another approach is based on large-scale semantic constructs, such as sections of text and the reference list. The restructuring of MIM took a third approach based on small-scale constructs. For example, instead of one structure for a whole reference list or a single reference, we broke down the references into components: authors, titles, journals, etc..

This paper describes the challenges and solutions encountered while making this evolutionary transition for the MIM database.

PROBLEMS AND GOALS

The previous major revision of MIM software resulted in the UNIX file system as the file repository and UNIX text editing as the main method for updating the entries. The problems of this system can be described from three perspectives: structure, maintenance, and access.

The structural problem relates to the fact that MIM entries were essentially flat text files and the software did not perform automatic structural validation. All errors (typographical and semantic) were only detected by proofreading. However, many errors were not found and accumulated until book publication. At this time, errors were discovered and corrected as the book production programs broke down while processing files with errors or when the pre-book comprehensive proofreading was done. While the flat files gave the most freedom to the writers in layout and presentation, they created a context where the information is organized only for textual editing and presentation. To provide automated validation, our solution was to use fine grain structure, so that structured contents could be verified against each other without natural language parsing. A goal for the new MIM system was to define the structure and then to enforce it. Furthermore, as the structure of MIM evolves with time, the enforcing mechanism must be simple to adapt and the conversion of existing entries to new schema must be simple to perform.

The MIM database maintenance problem is a result of its legacy nature. The software is dependent on specific UNIX directories, files, and accounts. For example, the checkin and checkout programs that transferred the files in and out of the repository were processing files in the home directories of specific UNIX users. As changes were made in the operating system and in staff personnel, increasingly complicated modifications to the software were required. In addition, all programs that access MIM information were developed in an adhoc fashion, with very little reuse of existing software, nor was the code written in a format that facilitated reuse. For example, the programs for the book typesetter cannot be used for generating the files for the IRX system. The previous MIM software was based on 4,000 lines of UNIX sh code, 2,000 lines of UNIX lex code, and 4,000 lines of C code. A second goal for the new system was to improve reuse, capacity, and capability without sacrificing adaptability. In addition, a framework of extensibility should be at the core of the system so that the MIM database can migrate to other architectures with ease.

Last, the delivery of MIM information to users was limited to the book form, the IRX-based OMIM, and the IRX-derived flat files for Internet access. It lacked the navigational features needed to interoperate with the many new genomic databases that have been recently deployed. In the recent years, the technologies for novel methods of accessing information over the Internet have matured: Wide Area Information Servers (WAIS), gopher, and World-Wide-Web (WWW) [7]. In addition, CDROM (multimedia) environments have also gained wide acceptance. MIM users were expecting to receive MIM information through these new methods. Therefore, our third and final goal was to ensure that the new MIM structure and software were designed to simplify the transporting of MIM information to new environments and to facilitate new ways of accessing MIM information.

A major underlying concern for all these goals was that the new MIM repository had to be of production quality and not a research system. Therefore, it had to be robust enough for heavy concurrent usage and provide sufficient throughput for on-demand access and editing sessions. In addition, three overriding principles were used when developing this new system: industry standards, open system architecture, and extensibility.

APPROACHES

We addressed the structural problem of MIM by tagging the elements in a MIM entry and then ensuring that the content of each tag was appropriate for the tag and the tags were in the correct order. The ISO 8879 standard, Standard Generalized Markup Language (SGML) [8], was used for such tagging or mark up of MIM documents. SGML provided a method of specification (Document Type Definition, DTD) that defines the structure of the document in terms of the tags and defines whether documents that have been tagged conform to a particular DTD. Because SGML is an international standard, many commercial and public domain software support the processing of SGML DTD's and tagged documents.

The software problem of MIM was best solved by using an open architecture, i.e., nonproprietary systems with standard application interfaces. This type of architecture allows vendor independence and future evolution. Unlike SQL, there are no standards widely accepted for text retrieval. In addition, we could not find a SGML-based textual database management system (DBMS) that was within our budget constraints. Consequently, we opted for in-house development, but made sure that it would be easy to migrate when an appropriate candidate DBMS is found. The software engineering tools and environment used for the Genome DataBase (GDB) [9] were adopted for MIM. These tools provided automatic set up, versioning, and common access for groups of users.

The solution for accessing MIM information in different distribution media was based on a set of filtering programs that convert the SGML-based MIM files to the appropriate format required for that particular medium. For example, if the typesetter can directly process SGML, then no conversion is needed. In fact, the eleventh edition of the MIM book was directly printed from SGML. The other formats (IRX, word-processing, and WWW) were given their own independent programs. This approach ensured the extensibility of the system for future methods of access.

Although the database implementation was done inhouse, the problem of editing MIM files remained. SGML has a set of rich, i.e., "complex," features to minimize the intrusion of "tags" in the flow of the document, thereby allowing rapid keying of SGML tagged content using text editors and minimizing storage space. Plain text editors are not acceptable as editing tools because MIM staff writers are not experts in SGML rules. Therefore, we needed a tool with automatic enforcement of tags and their order. As a result, for the staff writers, we used a commercial SGML editing package that actively enforces the DTD-driven structure and provides a graphical user interface similar to the popular word processors used in the PC domain.

RESULTS

SGML Encoding for MIM

SGML does not convey any semantic information associated with the tags. It is up to the application domain or program to determine the meaning of the tags, i.e., whether it encodes semantic or formatting information. Since we chose to specify the MIM structure in terms of the semantic content, the tags relate to the meaning of the enclosed content. For example, a reference in MIM: "Fitzsimmons, J. S.: Familial recurrence of achondroplasia. Am. J. Med. Genet. 22: 609-613, 1985." can be tagged as:

```
<REF REFNO="13">
<AUTHOR>Fitzsimmons, J. S.</AUTHOR>
<TITLE>Familial recurrence of achondroplasia
  </TITLE>
<JOURNAL>Am. J. Med. Genet.</JOURNAL>
<VOLUME>22</VOLUME>
<PAGE>609-613<PAGE>
<DATE>1985</DATE>
</REF>
```

The start and end tags are "<label>" and "</label>", respectively, and they enclose a specific construct denoted by "label." This approach for defining the DTD for MIM ensures independence from distribution media because all formatting, including rearrangement, is performed as an independent step. It also offers the best approach for migration to traditional databases. Since direct editing is through a commercial SGML-based editor, writers never have to manually enter the tags as text string, so the minimization features of the SGML tags were not used. This, in turn, simplified the code for the parse engine because the start and end tags of all SGML constructs were always matched.

The choice of using SGML to specify structure also facilitated the evolution of MIM entries. Since the conversion in January 1994, the schema for the references has been changed three times to reflect more accurately the actual information content. Each time, the DTD modifications were carried out with minimal effort and the supporting software was updated with similar ease.

Validation of MIM Entries

Although SGML encodes the parts of MIM entries, it is unable to validate the semantic correctness of the contents. On the other hand, semantic validation is an open-ended problem that requires natural language processing in a narrative text file. A compromise was made: citations in the text that point to references in the reference list are checked for correct matching. For example, "...reported by Fitzsimmons (1985)..." can be tagged as: "...reported by <CIT REFNO="13"> Fitzsimmons (1985)</CIT>...," but only if there exists exactly one reference whose sole author is "Fitzsimmons" and the date of publication was 1985. When an exact match is found, the REFNO attribute of CIT will be assigned the REFNO attribute of the corresponding REF. This feature provided a bonus for the WWW users, because the citations are then presented as hypertext links to the references.

Most citations in MIM text follow a uniform convention, thus they can be automatically detected and matched against the references. However, after further analysis, the convention has four possible outcomes: (1) valid citation that uniquely points to a reference, (2) ambiguous citation that points to more than one reference, (3) problem citation that points to a missing reference, and (4) possible citation that may point to a reference. The semantic validation program will find all cases and properly tag them for correction.

Repairing Errors

The conversion to SGML was performed for the MIM database on January 1, 1994. Afterward, all editing activity occurred under the SGML-based editor. Therefore, all potential problems in the text files had to be identified and marked for later correction. The heuristic used was very sensitive and, to our knowledge, did not miss any problems, but the tradeoff was a high false-positive rate. Out of 6,500 MIM entries and 50,000 references prior to conversion, only 3,500 entries and 35,000 references were converted without any identifiable problems. Excluding semantic problems, e.g., uncited and ambiguous citations, approximately 1,500 entries with 3,000 references were tagged as having potential structural problems. Out of these, only 500 entries with 1,000 references had real syntactic problems that required editing, e.g., a missing separator between author list and title in the original flat file which resulted in a "error" tag. The other 1,000 entries with 2,000 references were false-positives due to the sensitivity of our heuristics, e.g., the presence of a title delimiter inside the title because the original title contains it. Despite the high false-positive rate, the correction phase took an acceptable three person-months to perform. Overall, MIM had problems in 10 percent of the entries, but in terms of references, only 2 percent. This reflects the excellent proofreading skills of the MIM staff.

After the conversion to SGML and the correction of structural problems, we looked for semantic problems by examining the component fields of MIM. This was very useful for identifying inconsistent usage of a particular field. For example, all text tagged with JOURNAL was extracted, sorted, and printed for review. Initially, there were 1,800 distinct journal spellings, of which 120 were obvious variants, such as abbreviated vs. nonabbreviated names. However, about 100 variant spellings cannot be easily resolved and these will require a library literature search to correct.

These syntactic and semantic variants did not pose a problem in the flat-file version of MIM because the user, through natural language, can almost always determine the actual name of the journal or separate the authors from the title. However, these variants have to be corrected prior to cross-linking these references to references in the Medline database.

UNIX File System Repository

Since a suitable SGML-based DBMS couldn't be found, we had to develop a robust, in-house text database. The existing repository was based on the UNIX file system and the only problems were related to the software that managed this repository, as described above. Therefore, we adopted this architecture and merged it with the software engineering environment, but completely rewrote the software.

The previous system only dealt with three writers and, therefore, did not need robust concurrency control. In contrast, the number of active writers could now reach ten or more and robustness became critical. We used the UNIX atomic command "mkdir" [10] as a concurrency control semaphore to enter critical code sections that directly manipulate database files. For example, to lock a MIM entry for update or retrieval, the command "mkdir $LOCK/$sig" is executed, where "$LOCK" is a global lock directory, and "$sig" is a signature for a database resource, in this case, the MIM number. Only the first "mkdir" command will succeed in creating the signature directory. All other "mkdir" with the same signature will fail and have to wait (or abort) until the first user has completed the transaction and released the lock by executing "rmdir $LOCK/$sig". This had the additional benefit of creating a directory for temporary files and simplifying rollback of modifications.

Revision Control System (RCS) [11] was used as the method for maintaining file history, but its locking method was not used as write locks for the writers. The reason was that 6,500+ entries cannot be effectively maintained in one UNIX RCS directory. Thus, three levels of directories, each covering two digits in the 6-digit MIM number, were used. For example, the file for MIM entry 214355 is located in the directory $MIM/db/210000/214300/214355, along with all the status files relating to 214355. The variable "$MIM" specifies the root directory of the MIM database. All editing is performed in a separate directory, $MIM/edit, and the software manage the transfer of files between the two.

Instead of the traditional database transaction control mechanism, we retained the RCS approach of checkout/checkin. This was managed by UNIX shell scripts. Since its inception, the system has handled peak loads of 350 updates per work day, with 50 entries checked out at any one time, without problems. This peak activity is about 10 times higher than normal, based on statistics gathered over the last 2 years; it occurred during the time immediately post-conversion, when a major effort was made to correct errors.

The validation and processing of MIM files were performed by PERL scripts [12]. A total of 3,000 lines of UNIX sh code and 7,000 lines of PERL code comprise the new system software. Although the code size did not decrease from the previous system, the functionality had increased dramatically. Instead of spreading the code across sh, lex, and C, the new code relies only on sh and PERL. The objective is to simplify long-term software maintenance because expertise in fewer and higher-level languages is needed.

Converting SGML to Different Media

In order to ensure that the contents of an entry are tagged according to their semantic nature, we eliminated any text alignment formatting by removing all line breaks and extra spaces in the file prior to processing. Consequently, in order to distribute the MIM files to a particular medium, a filter program had to be built according to the requirements of the medium and generated the necessary formatting codes. The benefit from this strategy is that MIM files can be generated for any media without additional editing because it only has semantic SGML constructs. Another benefit is that formatting can be adjusted by the software and all media-specific MIM files can be made to reflect the new format almost immediately.

As it turns out, the increase in code required by the individual filter programs is minimal. For example, the filter programs for IRX, WWW, and word processing share large portions of code. The resulting design used a 400-line PERL core engine to parse the SGML-encoded text and another 500 lines specific to each filter program. Out of the 500 lines, however, only 200 have to be written de novo with the remainder copied from a basic template with minor modifications. Using the core facility, the filter program for WWW was completed within one week after the initial request.

DISCUSSION

This newly evolved MIM database was planned, developed, and executed based on the principles of industry standards, open systems, and extensibility. It took two person-years of software development time for the conversion to SGML-based structured text and the completely rewritten system software. Afterward, it took three person-months of corrections to resolve the remaining structural problems. The end result of this transition provided new features for navigating to external databases, allowed multimedia information, and improved the consistency of data. The advances for the support staff maintaining the MIM database include structured editing, simplified code maintenance, and better migration capability. We believe our approach of fine grain structuring with SGML and conversion to an open, extensible system has application for other flat-file, text-oriented databases. The principles used for MIM can also be used for porting other legacy systems.

The work reported here, the conversion to structured documents and the consolidation of software, is only one step among many that the MIM system will undertake. There are several projects under development. The first is the migration toward a relational administrative database, which will provide open architecture and standard. The second project is the compilation and cross-linking of references into a citation database to provide access to Medline abstracts. A third project is the search for an appropriate SGML-based text DBMS and retrieval engine that can make full use of the SGML structure.

Mendelian Inheritance in Man is one of the oldest active databases in the medical field. Its users range from researchers to clinicians and, now through the Internet, the general public. Part of its longevity is the result of providing the expected services to its users. As long as it continues to evolve and meet these challenges, MIM will remain valuable to scientists and the public. We hope that this most recent evolution of MIM will take it to the 21st century as a flagship database in the field of medicine and genetics.

REFERENCES

1. McKusick, V. A.: Mendelian Inheritance in Man. 10th Edition. Johns Hopkins Press: Baltimore. 1992.
2. Harman, D., Benson, D., Fitzpatrick, L., Huntzinger, R., Goldstein, C.: IRX: An Information Retrieval System for Experimentation and User Applications. Proc. RIAO 88 Conf., 840-848, 1988.
3. Lawton, J., Burks, C., Martinez, F.: Overview of the LiMB database. Nucleic Acids Research. 17:5885-5899, 1989.
4. Nassif, R., Mitchusson, D.:Issues and approaches for migration/cohabitation between legacy and new systems. Proc. ACM 93 SIGMOD. 471-474, 1993.
5. Leoffen, A.: Text Databases: A Survey of Text Models and Systems. ACM Sigmod Record 23(1): 97-106, 1994.
6. Salton, G., Allan, J., Buckley, C.: Automatic Structuring and Retrieval of Large Text Files. Communications of ACM. 37(2): 97-108, 1994.
7. Obraczka, K., Danzig, P. B., Li, S.-H.: Internet Resource Discovery Services. IEEE Computer. 26(9): 8-22, 1993.
8. Goldfarb, C. F.: The SGML Handbook. Oxford University Press: New York. 1992.
9. Li, P., Emmel, T. C., Campbell, J.: Virtual Development Environment-A database software engineering system based on UNIX tools. (in preparation).
10. Leffler, S. J., McKusick, M. K., Karels, M. J., Quarterman, J. S.: The Design and Implementation of the 4.3BSD UNIX Operating System. Addison-Wesley: Reading. 1989.
11. Tichy, W. F.: RCS-A system for Version Control. IEEE Software Practice and Experience. 15(7): 637-654, 1985.
12. Wall, L., Schwartz, R. L.: Programming perl. O'Reilly & Assoc.: Sebastopol. 1991.

ACKNOWLEDGEMENTS

The authors wish to thank Drs. V. A. McKusick, P. L. Pearson, K. Fasman, and Mr. C. Brunn for their support on this project. In addition, the system was refined with the advice and help of the MIM staff of C. Bocchini, P. Foster, and T. Hentges. Finally, many thanks to P. Foster for her editorial assistance. The work was funded by NIH 5P141HG00586-03 and DOE DE-FC02-91ER61230.